GUÍA PARA EL USO DE MATLAB PARTE 1

GUÍA DE USUARIO BÁSICO PARA MATLAB

El programa Matlab

MatLab (MATrix LABoratory) es un programa para realizar cálculos numéricos con vectores y matrices. Una de las capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones.

MatLab es un gran programa de cálculo técnico y científico, que tiene su propio lenguaje de programación. Dicho lenguaje, es una herramienta de alto nivel para desarrollar aplicaciones técnicas fáciles de utilizar.

Al arrancar MatLab se abre una ventana como la indicada en la Figura 1.

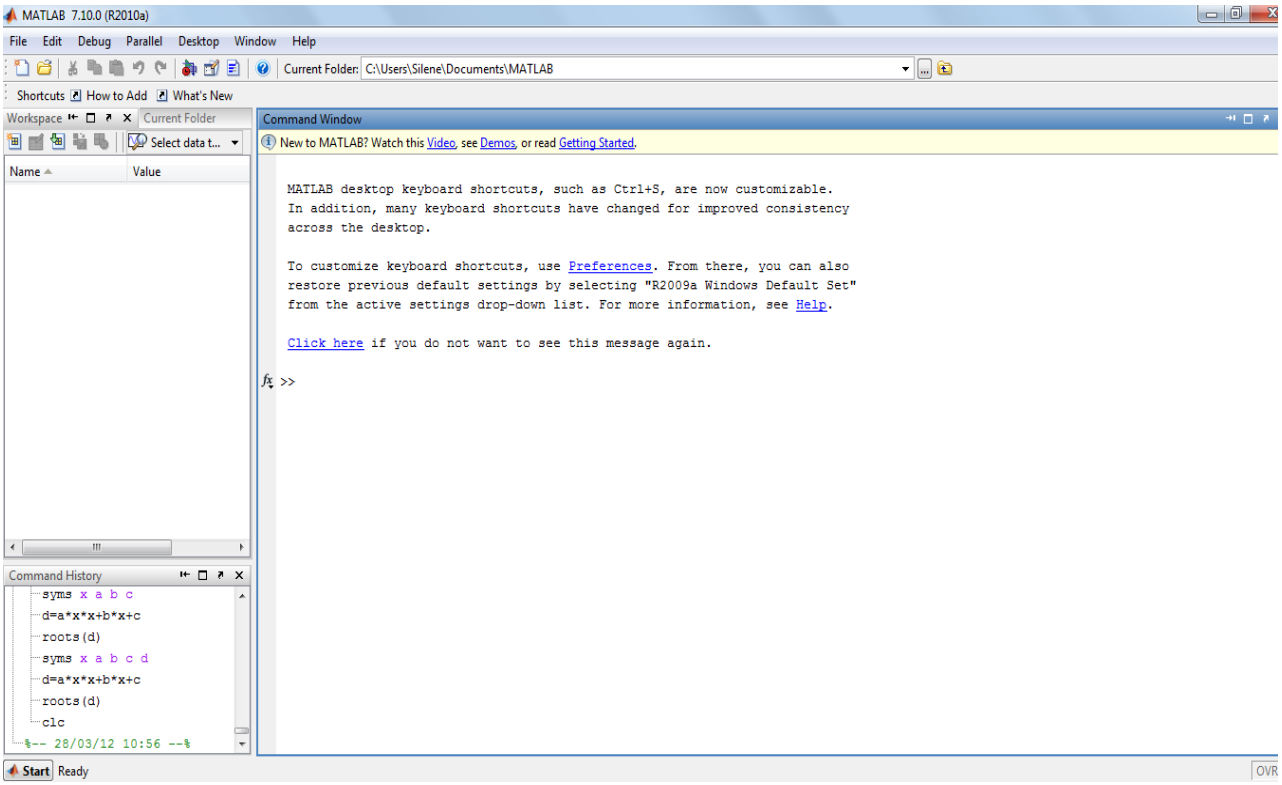


Figura 1 Ventana Inicial

La parte más importante de la ventana inicial es la *Command Windows*, que aparece en la parte derecha (recuadrada en rojo en la Figura 2). En esta sub-ventana es donde se ejecutan los comandos de MatLab, a continuación del símbolo característico (>>), que indica que el programa está preparado para recibir instrucciones.

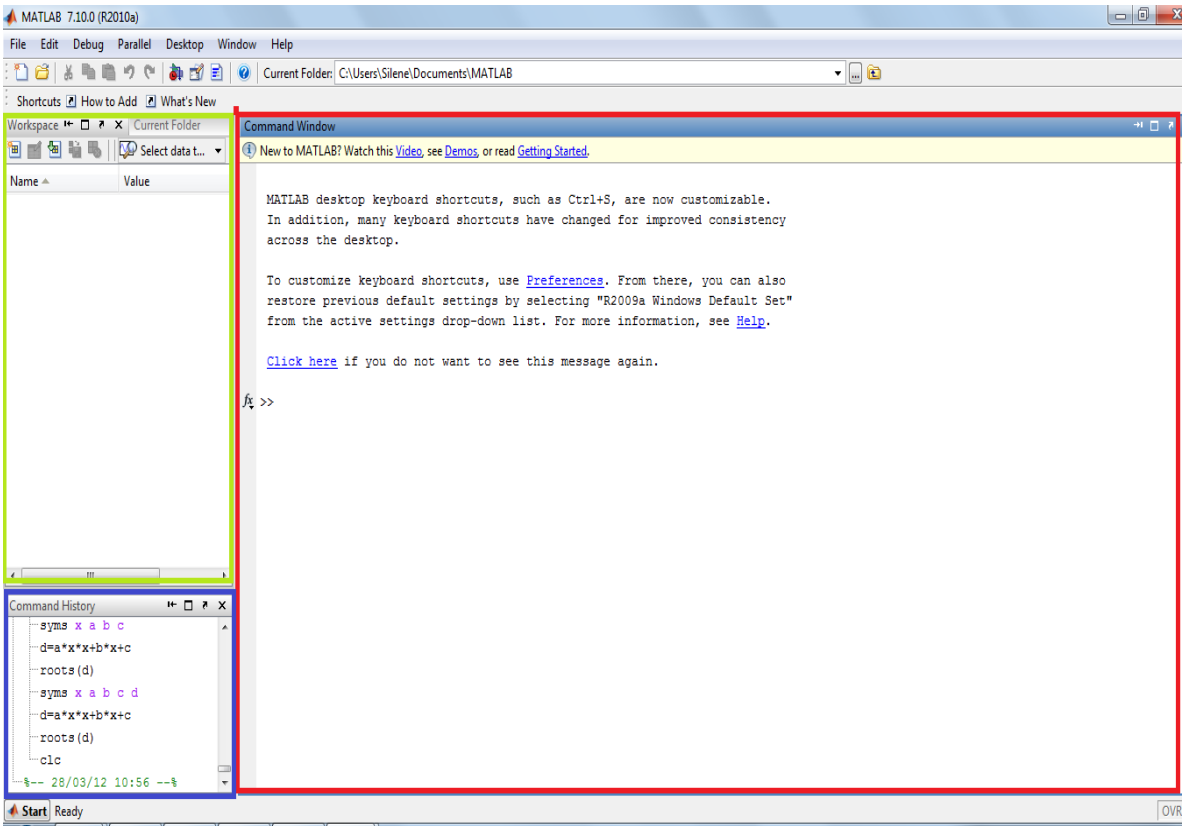


Figura 2 Sub-ventanas de comandos

En la parte izquierda de la pantalla aparecen dos ventanas también muy útiles: en la parte superior aparece la ventana (recuadrada en verde en la Figura 2) *Current Folder*, que se puede alternar con *Workspace* clickeando en la pestaña correspondiente. La ventana *Current Folder* muestra los ficheros del directorio activo o actual. El *Workspace* contiene información sobre todas las variables que se hayan definido en esta sesión y permite ver y modificar las matrices y vectores con los que se esté trabajando.

En la parte inferior aparece la ventana (recuadrada en azul en la Figura 2) *Command History* que muestra los últimos comandos ejecutados en la *Command Window*. Estos comandos se pueden volver a ejecutar haciendo doble click sobre ellos.

La instrucciones se pueden proporcionar directamente en la venta de comando (*Command Window*) o a través de ficheros-M (o M-files). Estos ficheros tienen la extensión *.m y contienen conjuntos de comandos o definición de funciones. La importancia de estos ficheros-M es que al teclear su nombre en la línea de comandos y pulsar Intro, se ejecutan uno tras otro todos los comandos contenidos en dicho fichero.

Aunque los ficheros *.m se pueden crear con cualquier editor de texto tal como Block de Notas, Notepad, Word, etc, MatLab dispone de un editor que permite crear y modificar estos ficheros, como ejecutarlos paso a paso para ver si contiene errores. La Figura 3 muestra la ventana principal del Editor. El editor muestra con diferentes colores los diferentes tipos o elementos constitutivos de los comandos (en verde los

comentarios, en rojo las cadenas de caracteres, etc.). Las líneas de comentario se indican con el carácter % delante del comentario.

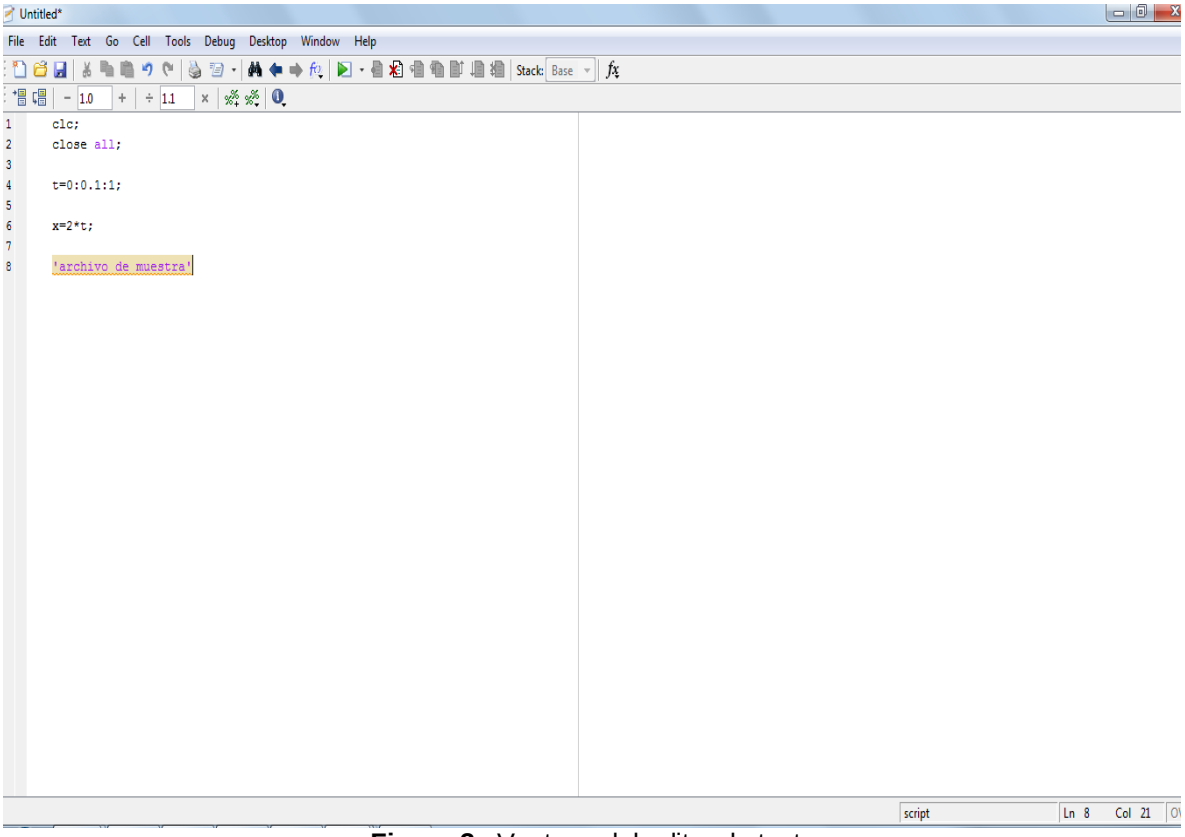


Figura 3 Ventana del editor de texto

Operaciones con vectores y matrices

Para definir una matriz o un vector no hace falta establecer de antemano su tamaño.

Las matrices se definen por filas, los elementos de la misma fila están separados por blancos o por comas, mientras que las filas están separadas por pulsaciones intro o por caracteres punto y coma (;). Por ejemplo el siguiente comando define una matriz 'A' de dimensión 3x3:

```
>> A=[1 2 3; 4,5,6;
7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9
```

De forma análoga a las matrices, es posible definir un vector fila ‘x’ en la forma siguiente:

```
>> x=[10 20 30]

x =

    10    20    30
```

En MatLab se accede a los elementos de un vector poniendo el índice entre paréntesis, por ejemplo:

```
>> x(2)

ans =

    20
```

Para definir un vector columna ‘y’ hay que separar los elementos por (;) o intro,

```
>> y=[11; 12; 13]

y =

    11
    12
    13
```

Y de la misma forma se accede a un elemento de él:

```
>> y(3)

ans =

    13
```

MatLab tiene en cuenta la diferencia entre vectores fila y vectores columna. Si intentamos sumar los vectores ‘x’ e ‘y’ obtendremos el siguiente mensaje de error:

```
>> x+y
??? Error using ==> plus
Matrix dimensions must agree.
```

Para poder sumar los vectores necesitamos que ambos sean vectores filas o ambos columna.

Si hacemos:

```
>> x'

ans =

    10
    20
    30
```

Transforma el vector fila ‘x’ en un vector columna.
Si sumamos ahora obtenemos:

```
>> x'+y

ans =

    21
    32
    43
```

Si transformamos ‘y’ en vector fila y sumamos, entonces:

```
>> x+y'

ans =

    21    32    43
```

Operaciones entre vectores

MatLab puede operar con vectores por medio de operadores y por medio de funciones.
Los operadores son los siguientes:

- Suma (+) :

```
>> x+y'

ans =

    21    32    43
```

- Resta (-) :

```
>> x-y'

ans =

    -1     8    17
```

- Multiplicación (*):

1. Vector fila por vector columna, el resultado es un número.

```
>> x*y

ans =

   740
```

2. Vector columna por vector fila, el resultado es una matriz.

```
>> y*x

ans =

   110   220   330
   120   240   360
   130   260   390
```

- Producto elemento a elemento (.*):

1. Por un número:

```
>> x.*3

ans =

    30    60    90
```

2. Por un vector elemento a elemento:

```
>> x.*y'

ans =

   110   240   390
```

- División elemento a elemento:

1. División de todos los elementos del vector por un número(./)

```
>> x./3

ans =

    3.3333    6.6667   10.0000
```

2. División de un número por todos los elementos de un vector (.\)

```
>> x.\3

ans =

    0.3000    0.1500    0.1000
```

- Elevar a una potencia elemento a elemento (.^)

```
>> x.^3

ans =

        1000        8000       27000
```

Si anteriormente no hemos definido el vector x se hace de la misma forma, es decir:

```
>> [1 2 3 4].^2

ans =

     1     4     9    16

>> [1 2 3 4].*[1 -1 1 -1]

ans =

     1    -2     3    -4
```

Funciones de Matlab

Funciones Matemáticas elementales: Se aplican a valores escalares o a vectores elemento a elemento.

sin(x) -> función seno
cos(x)-> función coseno
tan(x) -> función tangente
log(x) -> función logaritmo neperiano
log10(x) -> función logaritmo decimal
exp(x) -> función exponencial
sqrt(x) -> función raíz cuadrada
round(x) -> función redondeo hacia el entero mas próximo
abs(x) -> función valor absoluto

Funciones que actúan sobre vectores: Las siguientes funciones solo actúan sobre vectores.

[xm,im]=max(x)	Devuelve el valor máximo xm y la posición im del vector x
[ym,jm]=min(x)	Devuelve el valor mínimo ym y la posición jm
sum(x)	Suma de los elementos de un vector
mean(x)	Valor medio de los elementos del vector
std(x)	Desviación estándar

Funciones para cálculos con polinomios: Para MatLab un polinomio se puede definir mediante un vector de coeficientes. Por ejemplo, el polinomio:

x4-8x2+6x -10=0

se puede representar mediante el vector [1, 0, -8, 6, -10]. MatLab puede realizar diversas operaciones sobre él, como por ejemplo evaluarlo para un determinado valor de 'x'

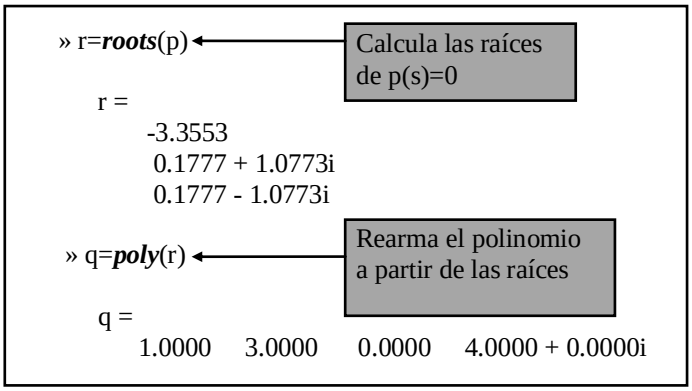
```
>> pol=[1 0 -8 6 -10]

pol =

     1     0    -8     6   -10
```

Si se tiene el siguiente polinomio:

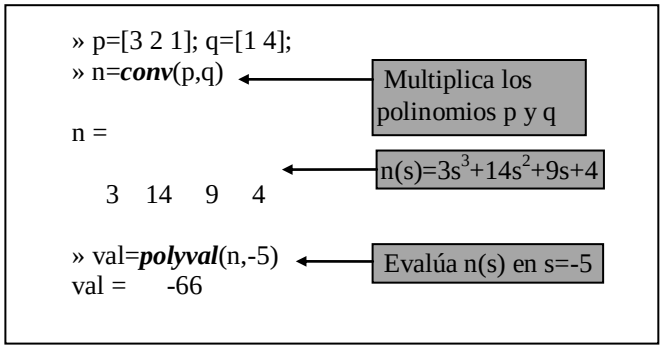
$$p(s) = s^3 + 3s^2 + 4$$

```
» p=[1 3 0 4]
```

Algunas de las funciones que podemos realizar sobre éste son las siguientes:



También podemos realizar:



Gráficas bi-dimensionales

MatLab dispone de cinco funciones básicas para crear gráficos 2-D. Estas funciones se diferencian principalmente por el tipo de escala que utilizan en los ejes de abscisas y de ordenadas. Estas cuatro funciones son las siguientes:

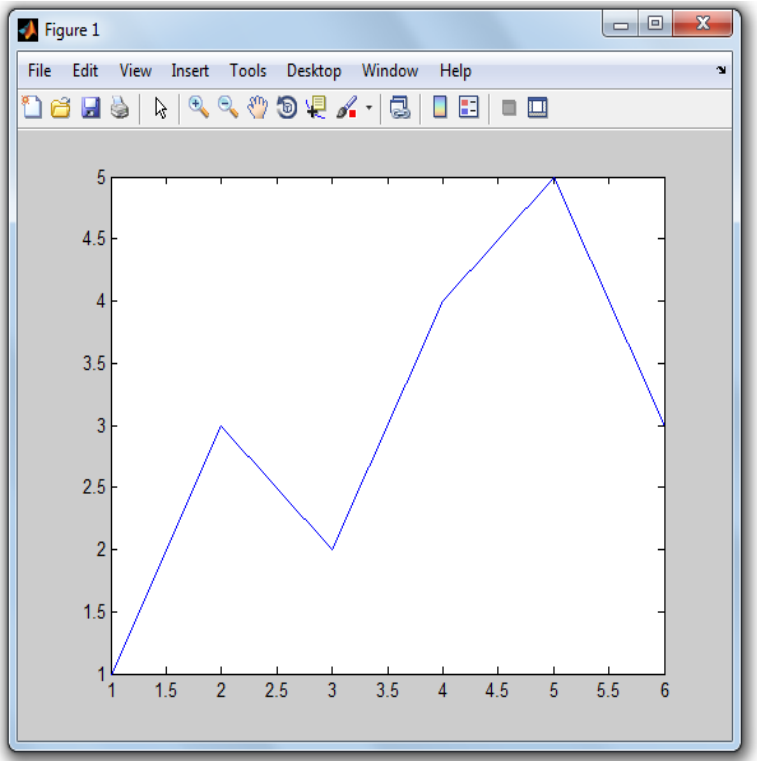
- **plot()** crea un gráfico a partir de vectores y/o columnas de matrices, con escalas lineales sobre ambos ejes.
- **loglog()** ídem con escala logarítmica en ambos ejes.
- **semilogx()** ídem con escala lineal en el eje de ordenadas y logarítmica en el eje de abscisas.
- **semilogy()** ídem con escala lineal en el eje de abscisas y logarítmica en el eje de ordenadas.
- **plotyy()** dibuja dos funciones con dos escalas diferentes para las ordenadas, una a la derecha y otra a la izquierda de la figura.

Existen además otras funciones orientadas a añadir títulos al gráfico, a cada uno de los ejes, a dibujar una cuadrícula auxiliar, a introducir texto, etc. Estas funciones son las siguientes:

- **title('título')** añade un título al dibujo.
- **xlabel('tal')** añade una etiqueta al eje de abscisas. Con 'xlabel off' desaparece.
- **ylabel('cual')** añade una etiqueta al eje de ordenadas. Con 'ylabel off' desaparece.

Ejemplo:

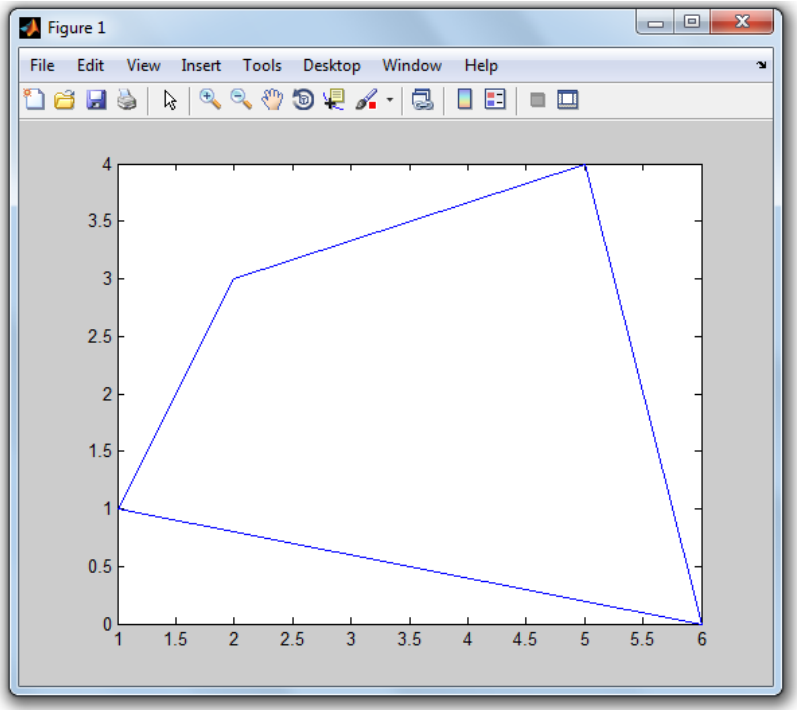
```
>> x=[1 3 2 4 5 3]

x =

     1     3     2     4     5     3

>> plot(x)
>>
```



Una segunda forma de utilizar la función plot() es con dos vectores como argumentos. En este caso los elementos del segundo vector se representan en ordenadas frente a los valores del primero, que se representan en abscisas.

Ejemplo:

```
>> x=[1 6 5 2 1]; y=[1 0 4 3 1];
>> plot(x,y)
>>
```



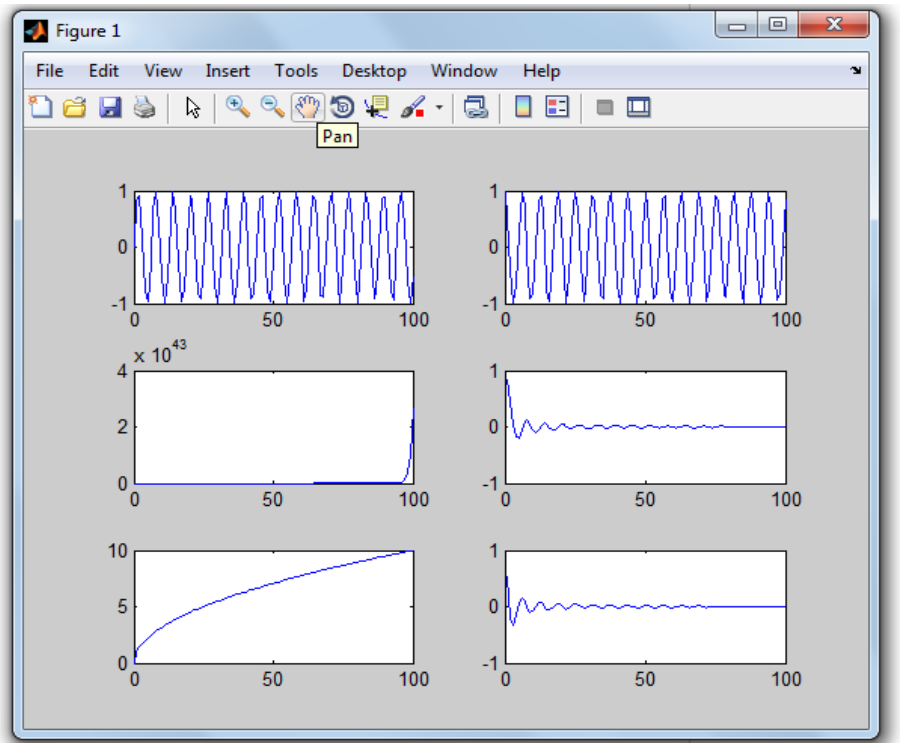
Por último, desde la ventana de graficación pueden realizarse cambios en las formas de mostrar las funciones como por ejemplo modificar colores, escalas, agregar leyendas, etc.

Una ventana gráfica se puede dividir en m particiones horizontales y n verticales, con objeto de representar múltiples gráficos en ella. Cada una de estas subventanas tiene sus propios ejes, aunque otras propiedades son comunes a toda la figura. La forma general de este comando es:

```
>> subplot(m,n,i)
```

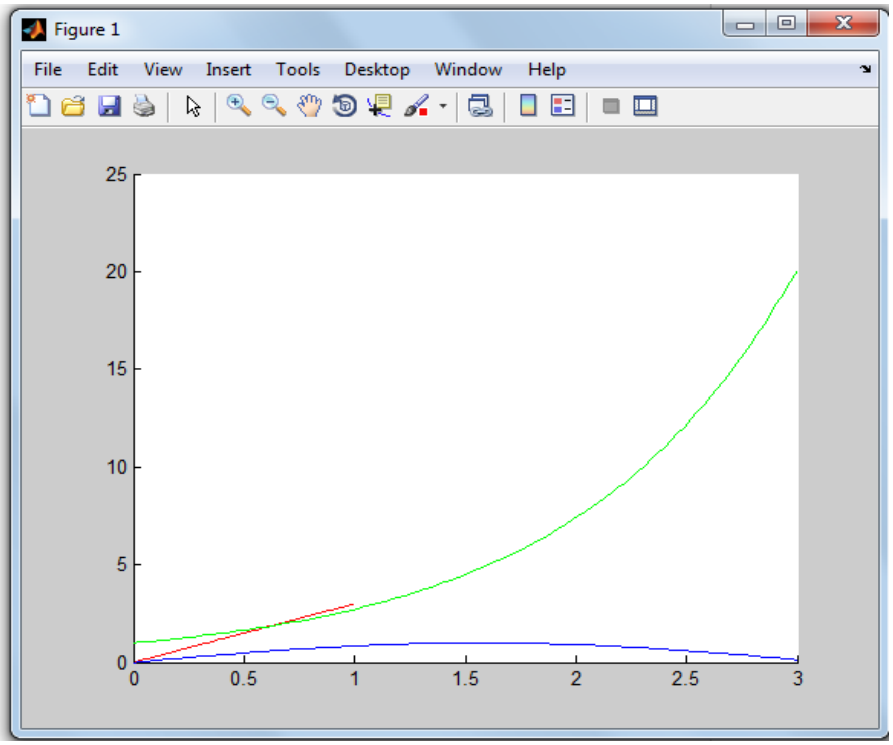
donde m y n son el número de subdivisiones en filas y columnas, e i es la subdivisión que se convierte en activa. Las subdivisiones se numeran consecutivamente empezando por las de la primera fila, siguiendo por las de la segunda, etc. Por ejemplo, la siguiente secuencia de comandos genera seis gráficos en la misma ventana:

```
x=0:1:100;
y=sin(x);
z=cos(x);
w=exp(x);
subplot(3,2,1)
plot(x,y)
subplot(3,2,2)
plot(x,z)
subplot(3,2,3)
plot(x,w)
subplot(3,2,4)
plot(x,y./x)
subplot(3,2,5)
plot(x,sqrt(x))
subplot(3,2,6)
plot(x,z./x)
```



Es posible dibujar varias funciones en una sola ventana con la instrucción:
>>hold on;
Para desactivarla solo es necesario escribir:
>>hold off;

```
t=0:0.01:1;
x=3*t;
y=sin(x);
p=exp(x);
hold on;
plot(t,x, 'r');
plot(x,y);
plot(x,p, 'g');
hold off;
```



GUIA PARA EL USO DE MATLAB PARTE 2
TOOLBOX DE CONTROL

FUNCION DE TRANSFERENCIA

MatLab es una potente herramienta para el análisis de 'sistemas' descriptos por funciones de transferencia.
La función de transferencia de un sistema lineal e invariante en el tiempo, relaciona la transformada de Laplace de la salida con la transformada de Laplace de la entrada en un sistema de ecuaciones diferenciales a condiciones iniciales nulas. En forma genérica se representa de la siguiente forma:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{a_m s^m + a_{m-1} s^{m-1} + ... + a_1 s + a_0}{s^n + b_{n-1} s^{n-1} + ... + b_1 s + b_0}$$

En sistemas reales o físicamente realizables m <= n.
El polinomio del denominador igualado a cero representa la ecuación característica que se utiliza ampliamente en el análisis de la estabilidad del sistema.

Para crear funciones de transferencia en MatLab se utilizan los siguientes comandos:

a) g=tf(num,den)

dónde "num" es un vector que contiene los coeficientes del polinomio del numerador de G(s) ordenado respecto a las potencias de s donde el primer elemento es el coeficiente que acompaña a la mayor potencia de s. "den" es otro vector que contiene los coeficientes del polinomio del denominador de G(s) ordenados de la misma forma que para el numerador.

Ejemplo de sintaxis en MatLab

$$G(s) = \frac{s^2 + 2.s + 3}{s^3 + 3.s^2 + 3.s + 1}$$

% Introducir una función de transferencia polinómica

```
>> num=[1,2,3];
>> den=[1,3,3,1];
>> G=tf(num,den)

Transfer function:
    s^2 + 2 s + 3
---------------------
s^3 + 3 s^2 + 3 s + 1
```

b) g=zpk(z,p,k)

Donde "z" es un vector que contienen los ceros del numerador de G(s), "p" es un vector que tiene los polos de G(s) y "k" es la ganancia estática de G(s)

Ejemplo de sintaxis en MatLab

%Cargar en Matlab una G(s) que tiene ceros en -1 y -2, polos en -10, -3+/-3i
% y ganancia estática k=5

$$G(s) = \frac{5(s+1)(s+2)}{(s+10)(s\text{^}2+6s+18)}$$

```
>> z=[-1,-2];
>> p=[-10,-3+3i, -3-3i];
>> h=zpk(z,p,5)

Zero/pole/gain:
    5 (s+1) (s+2)
----------------------
(s+10) (s^2 + 6s + 18)
```

c) s=tf('s')

A partir de esta instrucción de puede utilizar la "s" en las expresiones polinómicas de G(s) para que Matlab las interprete como funciones de transferencia.

Ejemplo de sintaxis en MatLab

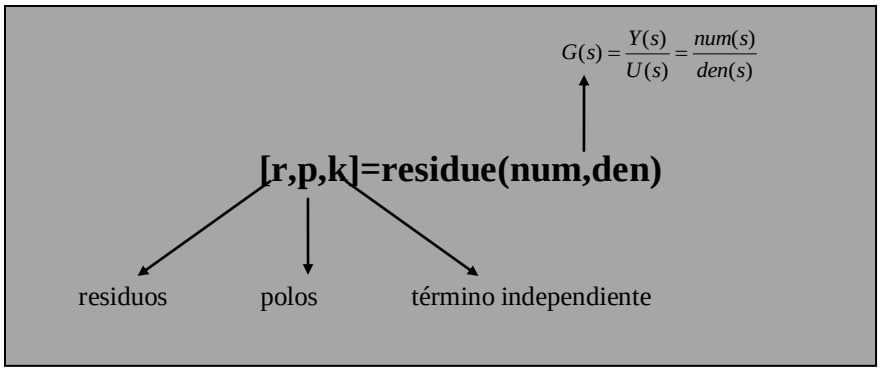
$$G(s) = \frac{s^2 + 2.s + 3}{s^3 + 3.s^2 + 3.s + 1}$$

% Introducir una función de transferencia polinómica

```
>> s=tf('s')

Transfer function:
s

>> G=(s^2+s*2+3)/(s^3+3*s^2+3*s+1)

Transfer function:
    s^2 + 2 s + 3
--------------------
s^3 + 3 s^2 + 3 s + 1
```

Una función de transferencia se puede descomponer en *fracciones simples*, utilizando la instrucción **'residue'** de MatLab. La diferencia con respecto a la forma tradicional de trabajo, es que, si existen polos complejos, los residuos que devuelve **'residue'** corresponden a cada uno de los polos:

$$G(s)=\frac{Y(s)}{U(s)}=\frac{a_m s^m+a_{m-1}s^{m-1}+\ldots+a_1 s+a_0}{s^n+b_{n-1}s^{n-1}+\ldots+b_1 s+b_0}=\frac{r(1)}{s-p(1)}+\frac{r(2)}{s-p(2)}+\ldots\ldots+\frac{r(n)}{s-p(n)}+k(s)$$



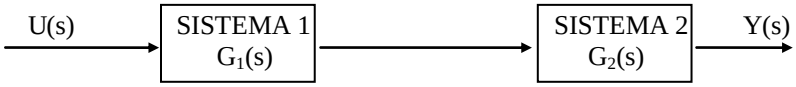
Como práctica ingresar la siguiente función de transferencia y descomponerla en fracciones simples :

$$G(s)=\frac{s+1}{s^3+3.s^2+s+7}$$

```
» n=[1 1];
» d=[1 3 1 7];
» [r,p,k]=residue(n,d)

r =

  -0.1630
   0.0815 - 0.1493i
   0.0815 + 0.1493i

p =

  -3.3307
   0.1654 + 1.4402i
   0.1654 - 1.4402i

k =     []
```

REPRESENTACIÓN EN DIAGRAMA DE BLOQUES:

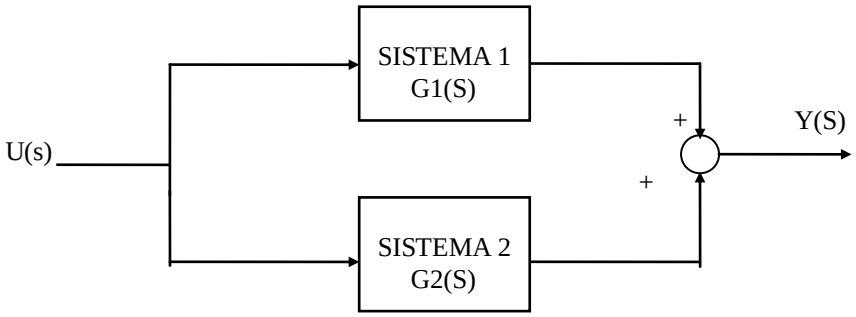
Los diagramas de bloques representan gráficamente a un sistema indicando las funciones realizadas por cada componente y el flujo de las señales. Generalmente los elementos que podemos encontrar en este tipo de representación son: la planta (elemento a controlar), controladores, actuadores, sensores, entre otros.
Un sistema sencillo de control a lazo abierto lo podemos obtener a partir de la interconexión en serie de la planta y del controlador. La función de transferencia obtenida para este caso en forma genérica sería:



$$T(s)=\frac{Y(s)}{U(s)}=\frac{num}{den} \qquad G1(s)=\frac{num1}{den1} \qquad G2(s)=\frac{num2}{den2}$$

[num,den]=series (num1,den1,num2,den2)

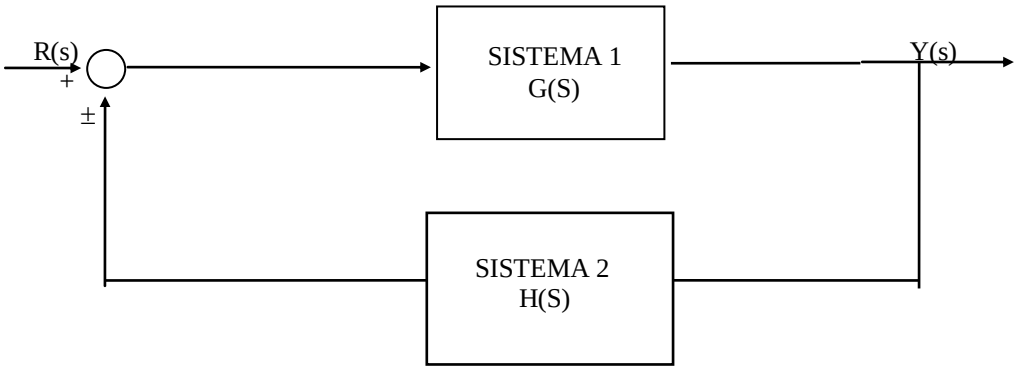
En ciertos sistemas pueden aparecer bloques en paralelo, y para su resolución, MatLab utiliza la instrucción **parallel**:



$$T(s)=\frac{Y(s)}{U(s)}=\frac{num}{den} \qquad G1(s)=\frac{num1}{den1} \qquad G2(s)=\frac{num2}{den2}$$

[num,den]=parallel(num1,den1,num2,den2)

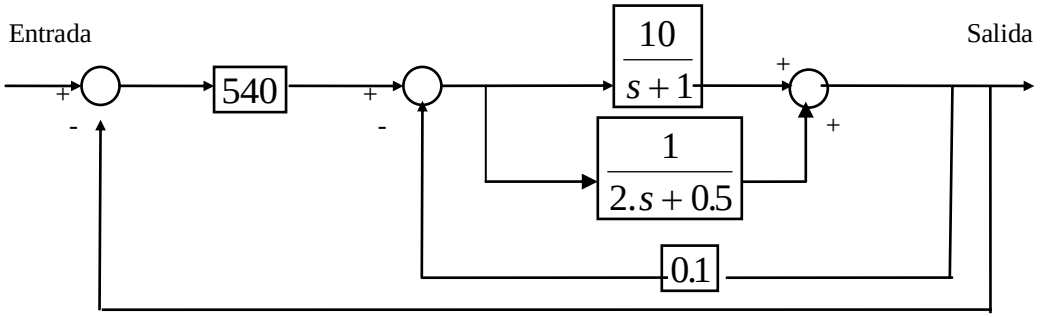
En el caso que se quiera obtener la función de transferencia de un sistema realimentado podemos utilizar la función **'feedback'** de MatLab para calcular la función resultante :



$T(s) = \frac{Y(s)}{U(s)} = \frac{num}{den}$ $G(s) = \frac{num1}{den1}$ $H(s) = \frac{num2}{den2}$ ±1 Realimentacion Positiva o negativa

[num,den]=feedback(num1,den1,num2,den2,signo)

Como ejemplo se calcula la función de transferencia total de los siguiente diagrama de bloques haciendo uso de las instrucciones de Matlab de dos maneras diferentes :

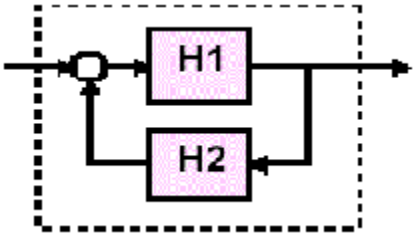
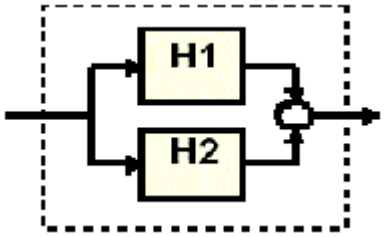
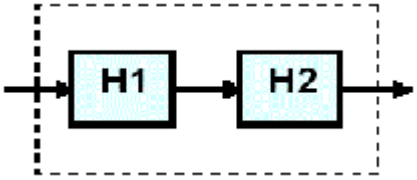



```
>> s=tf('s');
>> f1=10/(s+1);
>> f2=1/(2*s+0.5);
>> f3=0.1;
>> f4=540;
>> fp=parallel(f1,f2);
>> fb=feedback(fp,f3,-1);
>> fs=series(fb,f4);
>> ft=feedback(fs,1,-1);
>> ft
```

Transfer function:

11340 s + 3240
---
2 s^2 + 1.134e004 s + 3241

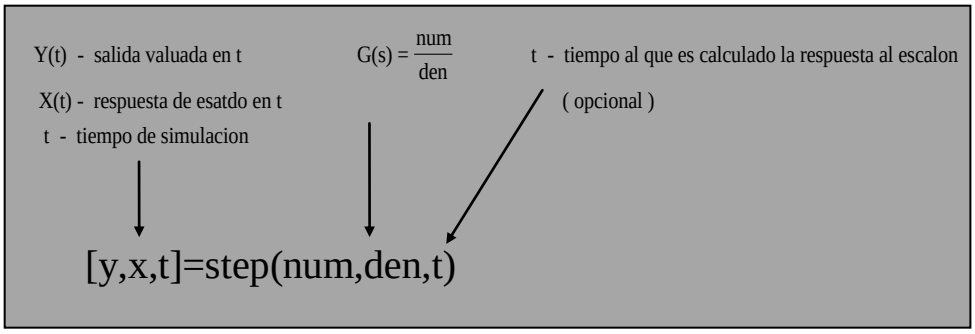
Resumen de instrucciones para la resolución de bloques:

Función en MATLAB	Descripción	Sintaxis basica	Diagrama
feedback	Sistema Realimentado	H=feedback(H1,H2)	
parallel	Es el paralelo de dos sistemas, es equivalente a sumar sistemas.	H=parallel(H1,H2)	
series	Sistemas en serie, es equivalente a multiplicar sistemas.	H=series(H1,H2)	

RESPUESTA TEMPORAL DE SISTEMAS:

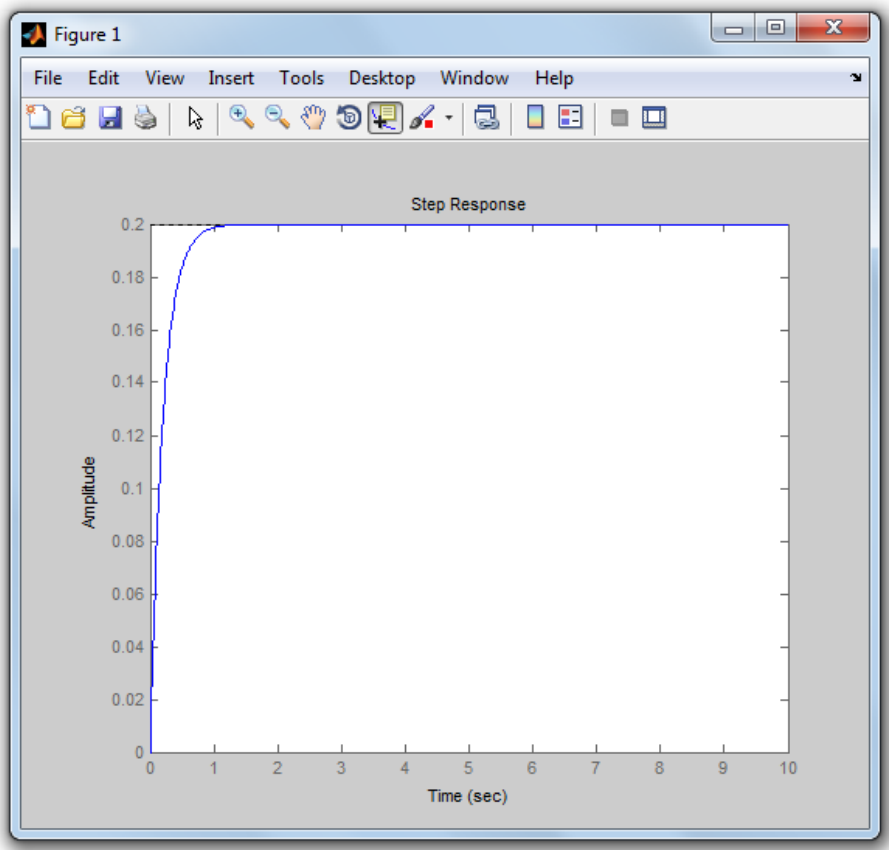
El objetivo que nos planteamos es el de mostrar cómo Matlab describe la respuesta temporal de cualquier sistema a diferentes entradas:

- **Entrada escalón :**

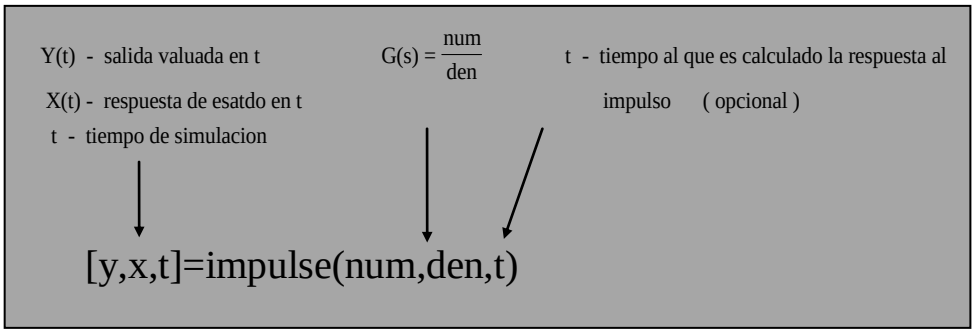


Ejemplo usando función de transferencia:

```
>> s=tf('s')

Transfer function:
s

>> g= 1/(s+5);
>> step(g,10)
>>
```

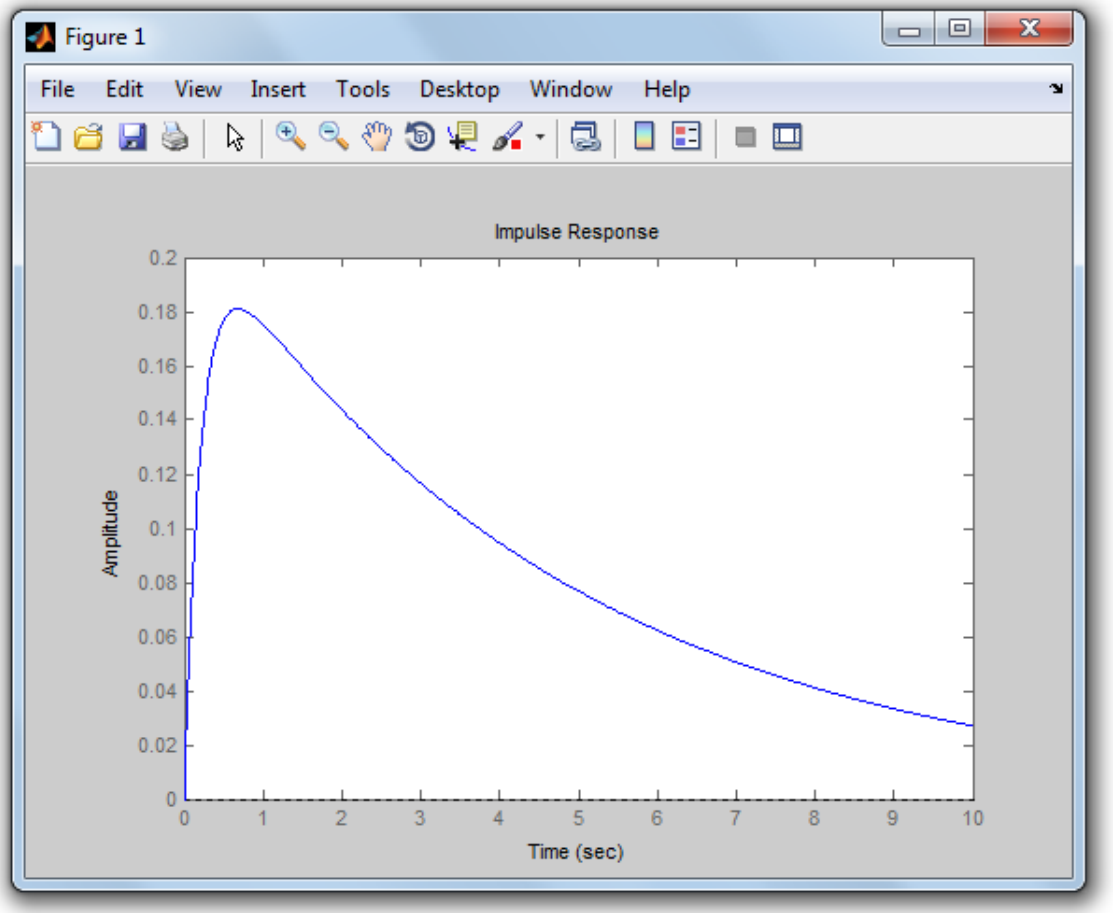


- **Entrada impulso :**

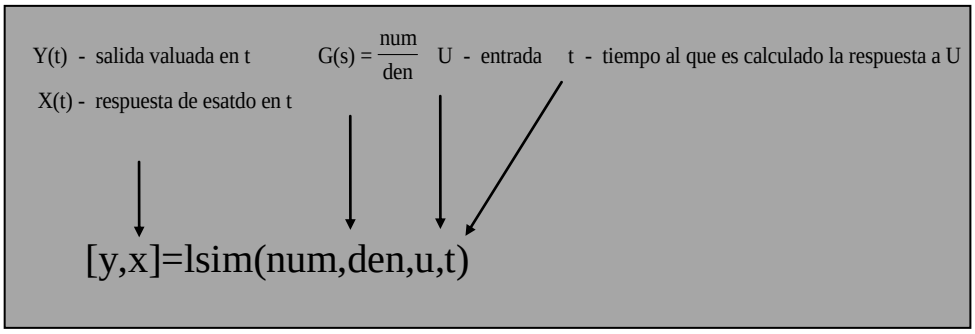


Ejemplo usando función de transferencia:

```
>> g= 1/(s*s+5*s+1);
>> impulse(g,10)
>>
```

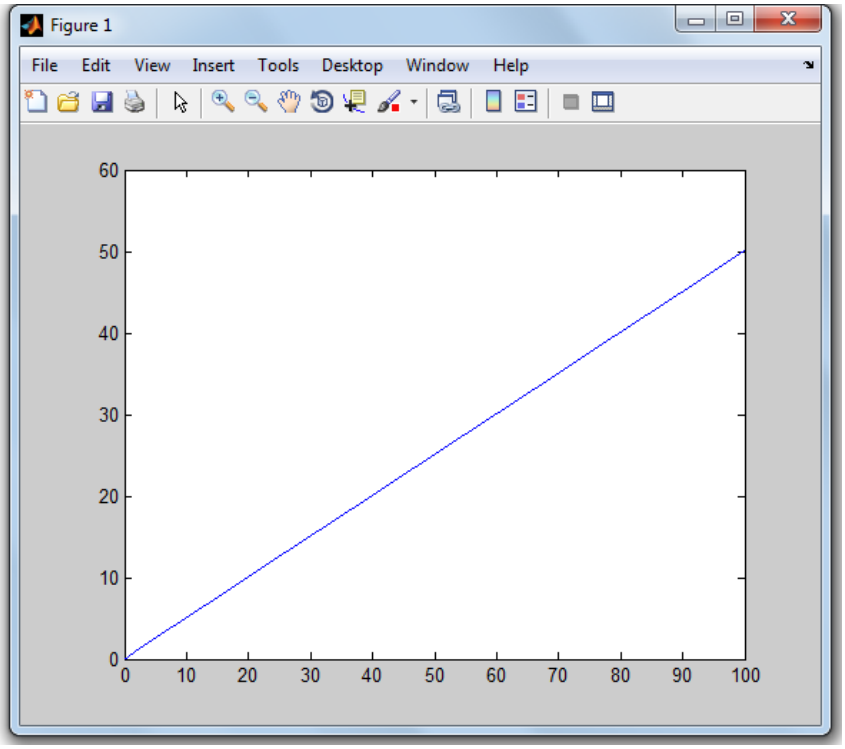


- **Entrada arbitraria creada por el usuario :**



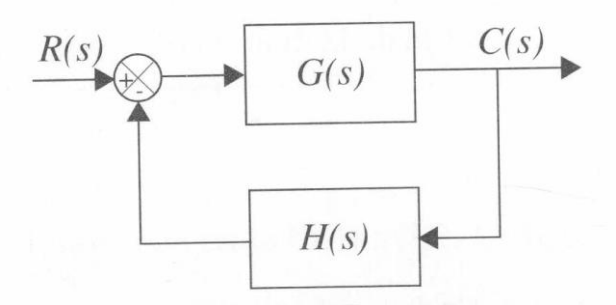
Ejemplo usando función de transferencia:

```
>> s=tf('s');
>> g=(s+1)/((s+2)*(s+3));
>> t=0:0.1:100;
>> u=t*3;
>> [y,x]=lsim(g,u,t);
>> plot(t,y)
>>
```



LUGAR DE RAÍCES:

Teniendo un sistema como el de la figura la función transferencia a lazo cerrado es:



$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)}$$

La ecuación característica esta dada por el denominador de esta función transferencia igualado a cero.

$$G(s)H(s) = -1$$

Para que esta igualdad se cumpla, y recordando que estamos en el plano complejo, se deben cumplir dos condiciones:

Condición de ángulo:

$$\angle G(s)H(s) = \pm 180°(2k + 1), \qquad k = 0, 1, 2,$$

Condición de módulo:

$$|G(s)H(s)| = 1$$

El lugar de raíces es un método a través del cual se puede visualizar como varían las raíces de la función transferencia a bucle cerrado a medida que la ganancia K del sistema crece de cero a infinito. Teniendo en cuenta que la ecuación característica debe tener la siguiente forma:

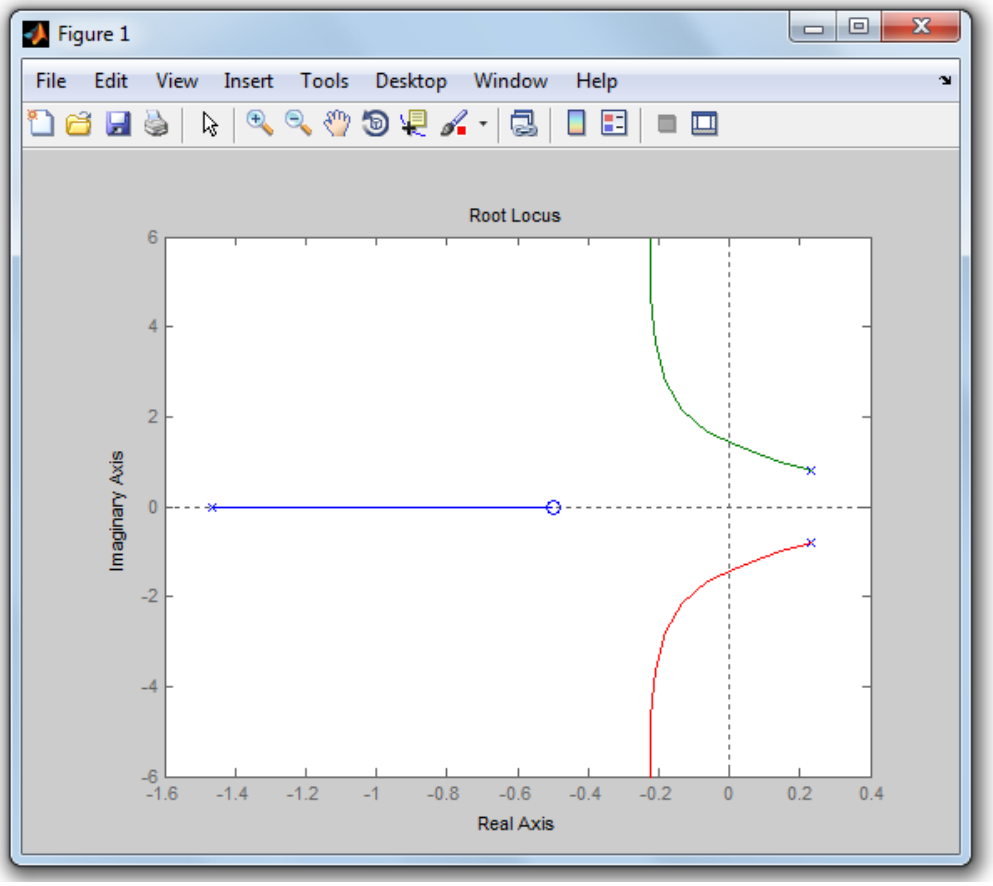
$$1 + \frac{K(s + z_1)(s + z_2)\ldots(s + z_m)}{(s + p_1)(s + p_2)\ldots(s + p_n)} = 0$$

Utilizando la instrucción '**rlocus**' en MatLab puede obtenerse el lugar de raíces de una función transferencia.

Ejemplo:

% Siendo g la función en transferencia a bucle abierto.

```
>> g=(s+0.5)/(s*s*(s+1)+1)

Transfer function:
   s + 0.5
-------------
s^3 + s^2 + 1

>> rlocus(g)
>>
```



Los detalles de cómo trabajar con la gráfica (límites, títulos, colores, etc) serán explicados en la sección de respuesta en frecuencia ya que son similares a los que se pueden realizar sobre los diagramas de Bode.

RESPUESTA EN FRECUENCIA:

La función respuesta en frecuencia representa la manera en que el sistema responde a entradas de diferentes frecuencias. Para calcularla, se reemplaza la variable ‘s’ de la función de transferencia por ‘jω’.

Bode

La orden ‘**bode**’ calcula las magnitudes y los ángulos de fase de la respuesta en frecuencia de sistemas continuos, lineales e invariantes en el tiempo. Los diagramas de Bode se utilizan frecuentemente para analizar y diseñar sistemas de control. Estos diagramas indican el margen de ganancia, el margen de fase, la ganancia, el ancho de banda, etc.

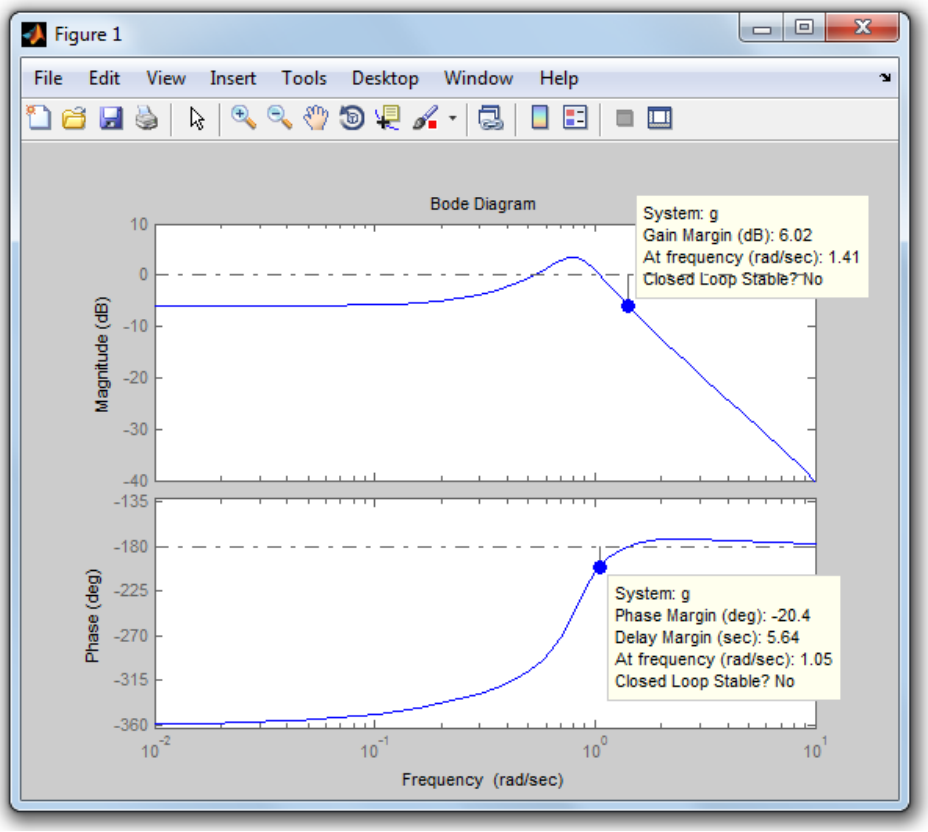
Cuando se introduce la orden '**bode**' (sin argumentos del lado izquierdo), MatLab realiza el diagrama de bode sobre la pantalla, permitiendo además recuperar datos de los diferentes puntos.

El diagrama de bode, resulta de mucha utilidad a la hora de establecer la estabilidad relativa de los sistemas. Para esto se utilizan los márgenes de estabilidad:

Margen de Ganancia: Es valor de la ganancia en db cuando la fase cruza por -180° cambiado de signo. Para la estabilidad, este parámetro debe ser mayor a 6 db.

Margen de Fase: Es el valor que toma la fase cuando la Magnitud cruza por 0db sumado a 180. Para la estabilidad, este parámetro debe ser mayor a 30°.

```
>> g

Transfer function:
   s + 0.5
-------------
s^3 + s^2 + 1

>> bode(g)
>>
```

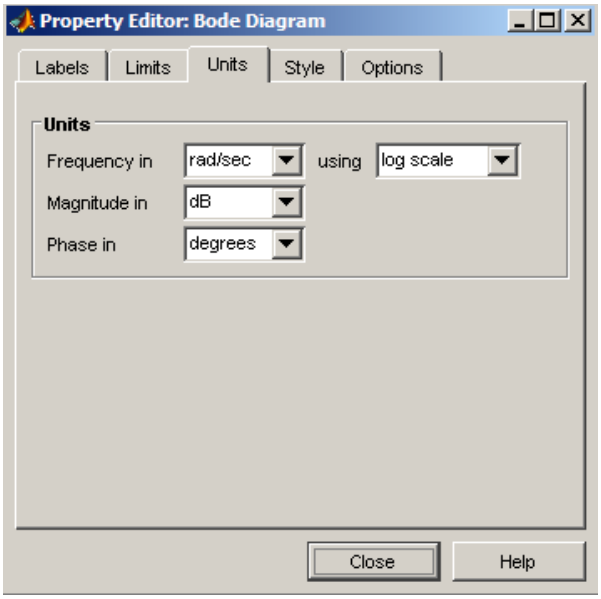


En la gráfica se muestran los valores correspondientes al margen de fase y al margen de ganancia del sistema. Para que éstos indicadores aparezcan en pantalla, se debe hacer click con el botón derecho del mouse sobre la gráfica, seleccionar la opción 'Características' y luego elegir 'Márgenes de estabilidad mínimos'. Hacer click con el botón izquierdo del mouse sobre los puntos para que aparezca el letrero.
Al hacer click con el botón derecho del mouse sobre las gráficas aparecen varias opciones:

- System: indica con qué color se están representando cada función de transferencia en caso de que estemos graficando varias de ellas. De igual

modo, se puede seleccionar para que solo aparezca en la gráfica la función de transferencia que a mí me interesa ver.

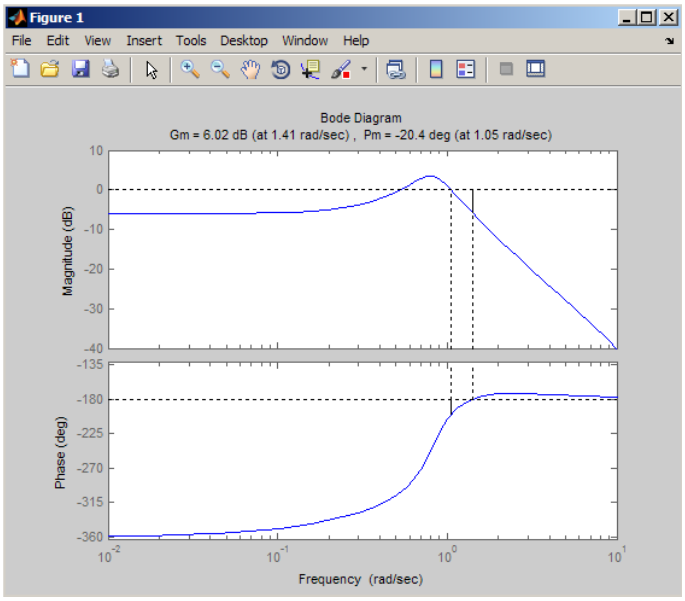
- Características: Posee tres opciones, de las cuales los dos últimas tiene relación a los márgenes de ganacia y fase. La primer opción, permite encontrar un pico en la magnitud en caso de que la gráfica lo posea.
- Show: Permite seleccionar si se quiere visualizar la magnitud, la fase o ambas gráficas en la misma ventana.
- Grid: Agrega una grilla que facilita la observación de las gráficas.
- Full view: Optimiza la visualización de las gráficas a fin de utilizar todo el espacio disponible con el trazado de la gráfica.
- Propiedades:



Dicha ventana permite modificar varios parámetros de la gráfica como por ejemplo: etiquetas sobre los ejes, límites de la grafica, unidades de las variables, estilos de gráfica (semilogarítmico, lineales) y desde esta ventana también se pueden varias las opciones antes descriptas.

Para obtener información de la grafica o de cualquier punto de la misma, se debe hacer click con el botón izquierdo sobre la gráfica. A continuación aparecerá un punto cuadrado negro, con la información de dicho punto. Si colocamos el mouse sobre el punto y mantenemos apretado el botón izquierdo, podemos arrastrar el punto sobre la gráfica hasta cualquier punto de interés de la misma.

Al escribir la sentencia **'margin (función de transferencia)'** se obtiene lo siguiente:



Por otra parte, es posible obtener los márgenes de ganancia y los márgenes de fase mediante la función 'margin'.

[mg,mf,wmg,wmf]=margin(num, den);

Mg es el margen de ganancia del sistema.
Mf es el margen de fase.
Wmg es la frecuencia a la cual sucede el cruce de la fase por -180º.
Wmf es la frecuencia de cruce de ganancia por cero.

Otra forma de introducir la sentencia bode en Matlab es la siguiente:

```
[mag,fase,w] = bode(num,den,w)
```

Las matrices mag y fase contienen las magnitudes y los ángulos de fase de la respuesta en frecuencia del sistema evaluados en los puntos de frecuencia especificados por el usuario.
El vector w se define, por ejemplo, de la siguiente manera:

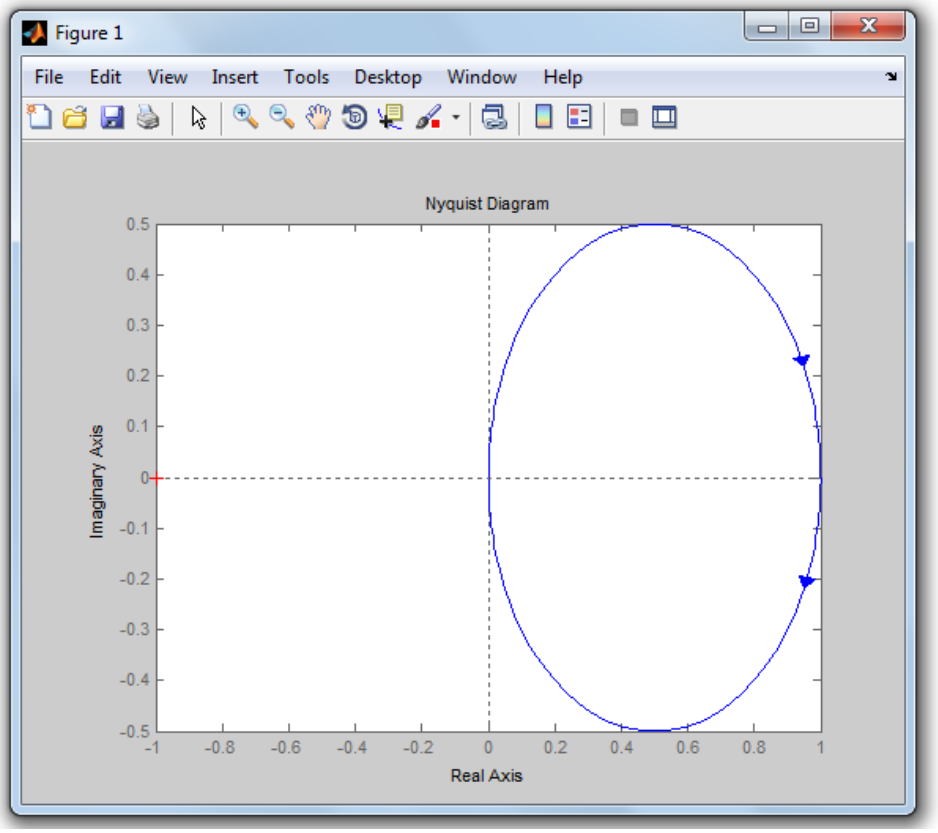
```
w = logspace(-2,3,100)
```

Siendo el valor representado por -2, en este caso, la potencia del límite menor de la gráfica. El 3 representa la potencia del límite superior. Por último, el 100 es la cantidad de espacios separados logarítmicamente entre los extremos.

Nyquist

Es otra manera de representar la respuesta en frecuencia del sistema. Consiste en una gráfica donde el eje 'x' se denomina eje real y el 'y', eje imaginario. La imagen se construye con la magnitud y la fase según cómo varíen con la frecuencia ω.

```
>> g=1/(s+1);
>> nyquist(g)
>>
```



Por otra parte, es posible obtener los márgenes de ganancia y los márgenes de fase mediante la función 'margin'.

[mg,mf,wmg,wmf]=margin(num, den);

Mg es el margen de ganancia del sistema.
Mf es el margen de fase.
Wmg es la frecuencia a la cual sucede el cruce de la fase por -180º.
Wmf es la frecuencia de cruce de ganancia por cero.